- **Create a simple RESTful service in Spring Boot which returns the Response "Welcome to spring boot".**

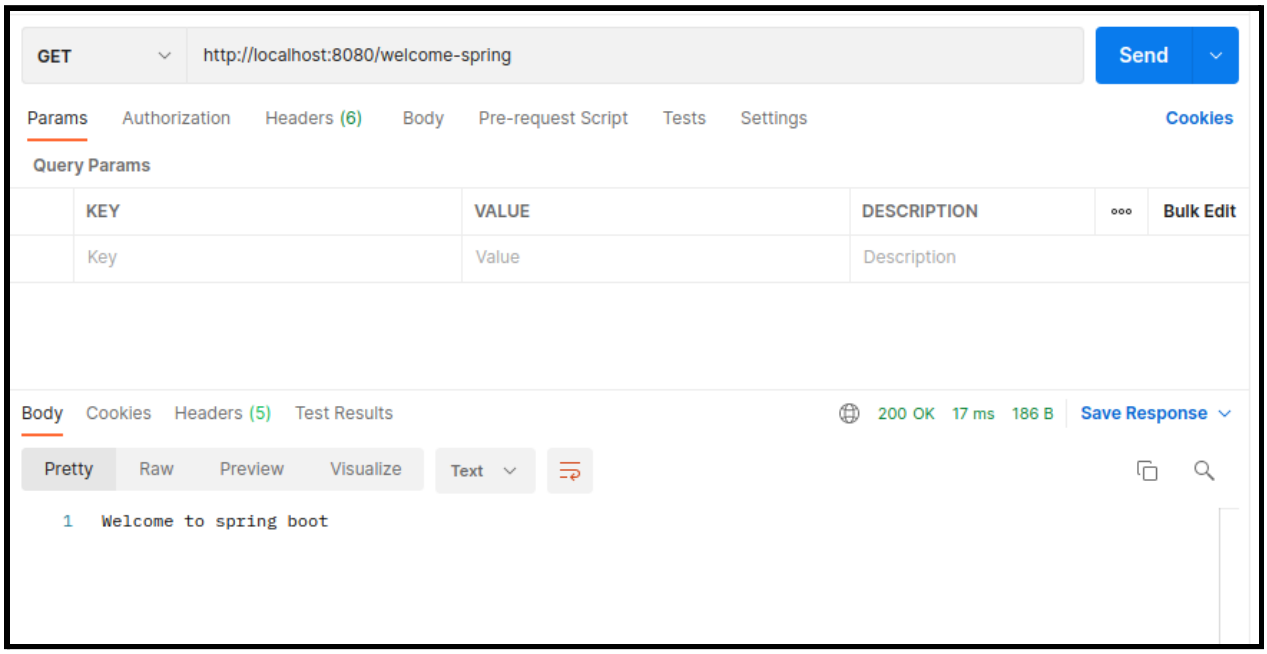
  <WelcomeToSpring.java>

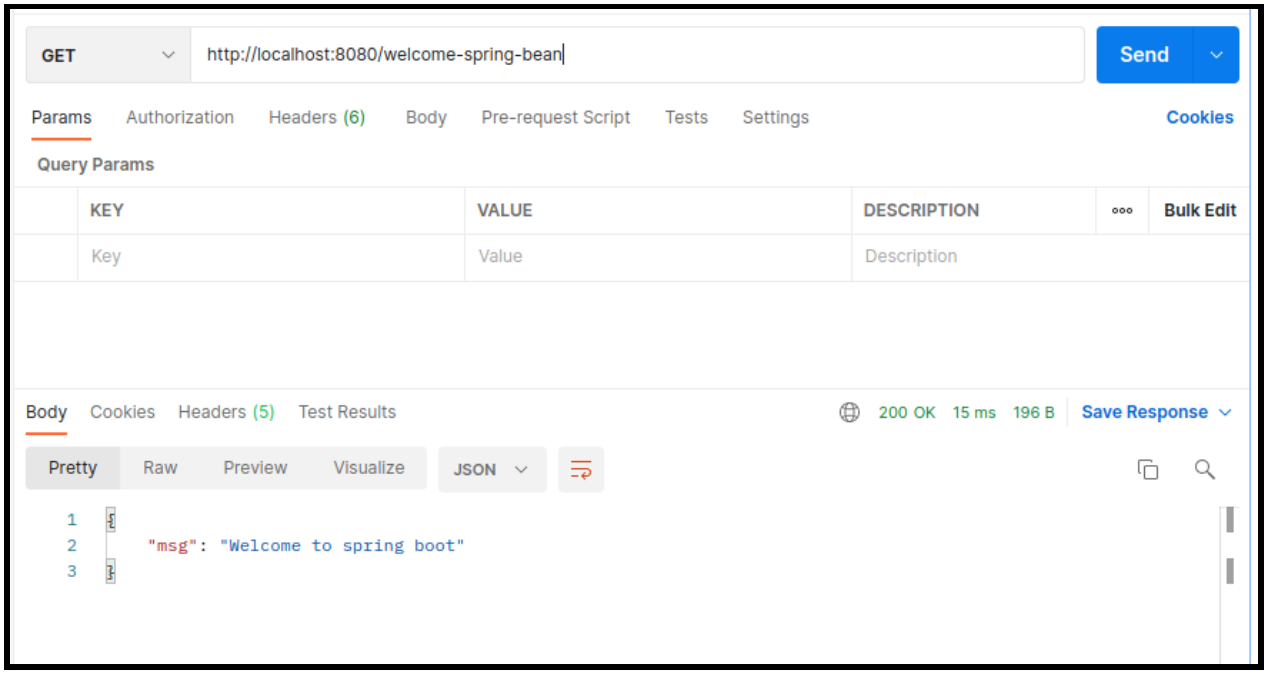
```
package com.ttn.spring.springRest.webServices.RestWebServices1.springWelcome;


public class WelcomeToSpring {
  private String msg;

  public WelcomeToSpring(String msg) {
     this.msg = msg;
  }

  public String getMsg() {
     return msg;
  }

  public void setMsg(String msg) {
     this.msg = msg;
  }
}
```

  <WelcomeToSpringController.java>

```
package com.ttn.spring.springRest.webServices.RestWebServices1.springWelcome;

import
com.ttn.spring.springRest.webServices.RestWebServices1.springWelcome.WelcomeTo
Spring;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class WelcomeToSpringController {

  @GetMapping("/welcome-spring")
  public String SpringWorld(){
     return "Welcome to spring boot";
  }

  @GetMapping("/welcome-spring-bean")
  public WelcomeToSpring springBootWorld(){
```

```
        return new WelcomeToSpring("Welcome to spring boot");
    }
}
```





- **Create an Employee Bean(id, name, age) and service to perform different operations related to employees.**

<EmployeeBean.java>

```
package com.ttn.spring.springRest.webServices.RestWebServices1.employee;

public class EmployeeBean {
  private Integer id;
  private String name;
  private Integer age;

  public EmployeeBean(Integer id, String name, Integer age) {
     this.id = id;
     this.name = name;
     this.age = age;
  }

  public Integer getId() {
     return id;
  }

  public void setId(Integer id) {
     this.id = id;
  }

  public String getName() {
     return name;
  }

  public void setName(String name) {
     this.name = name;
  }

  public Integer getAge() {
     return age;
  }

  public void setAge(Integer age) {
     this.age = age;
  }
```

```
  @Override
  public String toString() {
     return "EmployeeBean{" +
           "id=" + id +
           ", name='" + name + '\'' +
           ", age=" + age +
           '}';
  }
}
```

<EmployeeDaoService.java>

```
package com.ttn.spring.springRest.webServices.RestWebServices1.employee;

import org.springframework.stereotype.Component;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

@Component
public class EmployeeDaoService {
  private static List<EmployeeBean> employees = new ArrayList<>();
  private static int employeeCount = 3;

  static {
     employees.add(new EmployeeBean(1,"Vaishali",24));
     employees.add(new EmployeeBean(2,"Nidhi",30));
     employees.add(new EmployeeBean(3,"Sreyasi",23));
  }

  public List<EmployeeBean> findAll(){
     return employees;
  }

  public EmployeeBean save(EmployeeBean employee){
     if(employee.getId() == null)
        employee.setId(++employeeCount);
     employees.add(employee);
     return employee;
  }

  public EmployeeBean findOne(int id){
     for(EmployeeBean employee:employees){
```

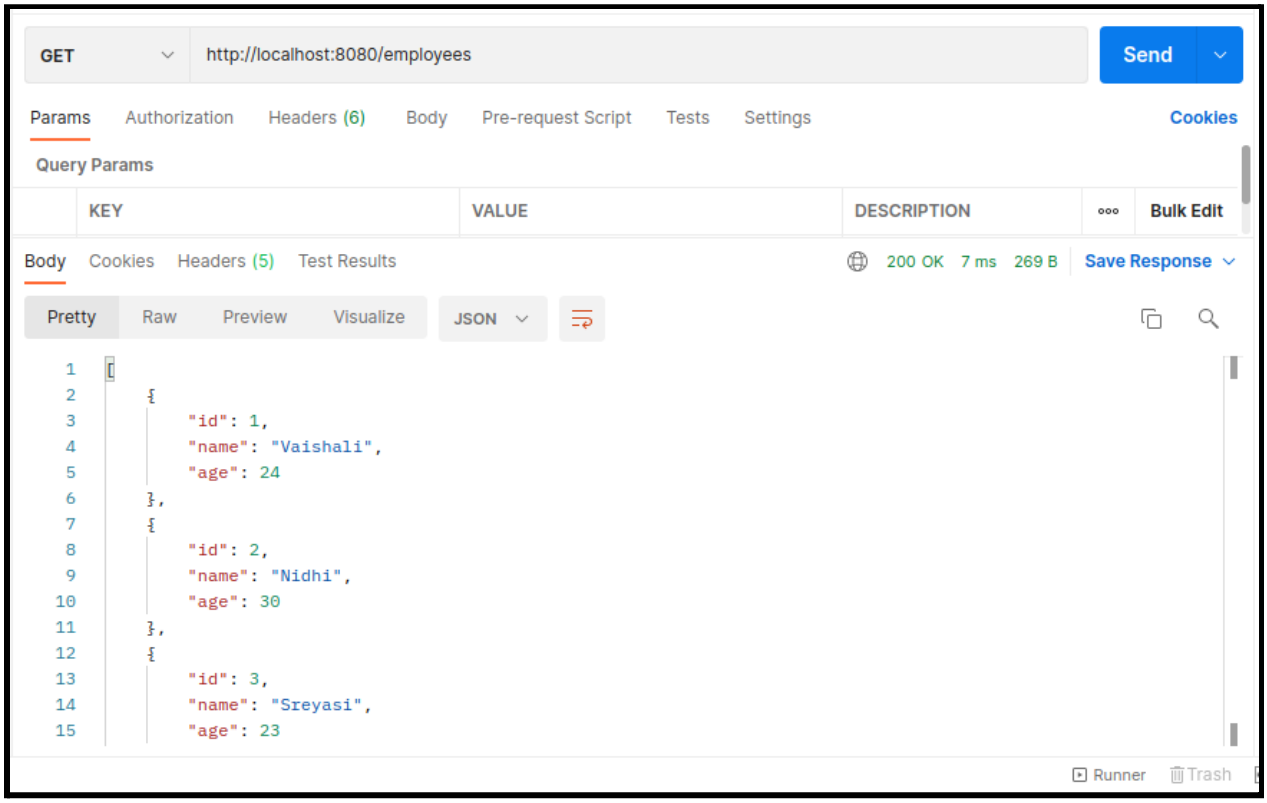
```
            if(employee.getId() == id)
                return employee;
        }
        return null;
    }

    public EmployeeBean deleteById(int id){
        Iterator<EmployeeBean> iterator = employees.iterator();
        while(iterator.hasNext()){
            EmployeeBean employee = iterator.next();
            if(employee.getId() == id){
                iterator.remove();
                return employee;
            }
        }
        return null;
    }
}
```

- **Implement GET http request for Employee to get list of employees.**

  <EmployeeController.java>

```
package com.ttn.spring.springRest.webServices.RestWebServices1.employee;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
public class EmployeeController {

    @Autowired
    private EmployeeDaoService service;

    @GetMapping("/employees")
    public List<EmployeeBean> retrieveAllEmployees() {
        return service.findAll();
```

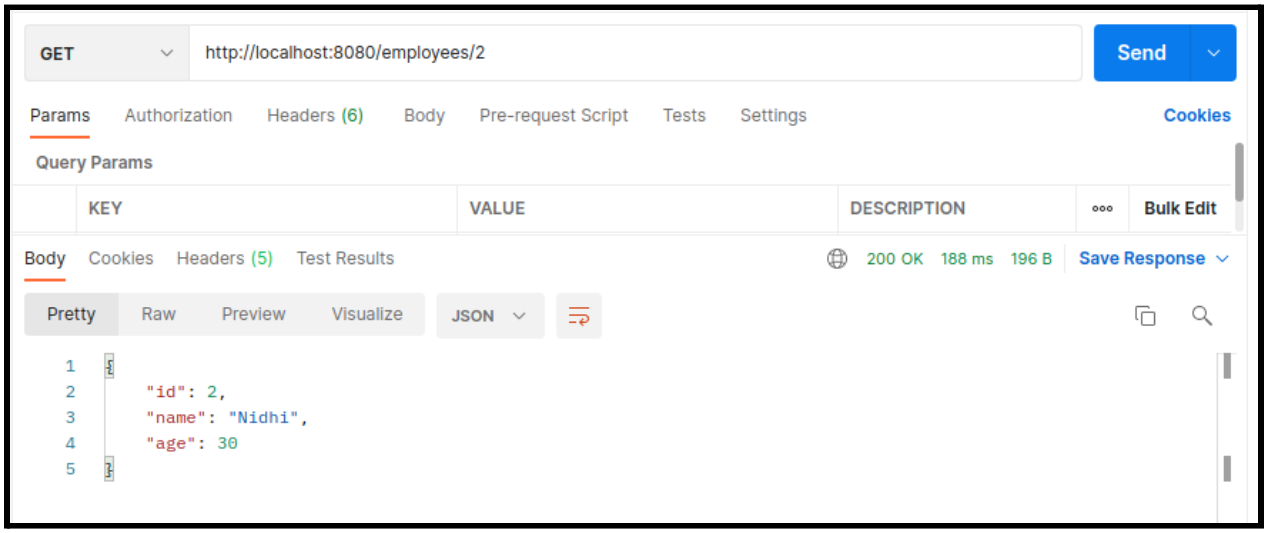
```
    }
}
```



- **Implement GET http request using path variable to get one employee.**

  <EmployeeController.java>

```
@GetMapping("/employees/{id}")
public EmployeeBean retrieveEmployee(@PathVariable int id){
   EmployeeBean employee = service.findOne(id);
   return employee;
}
```



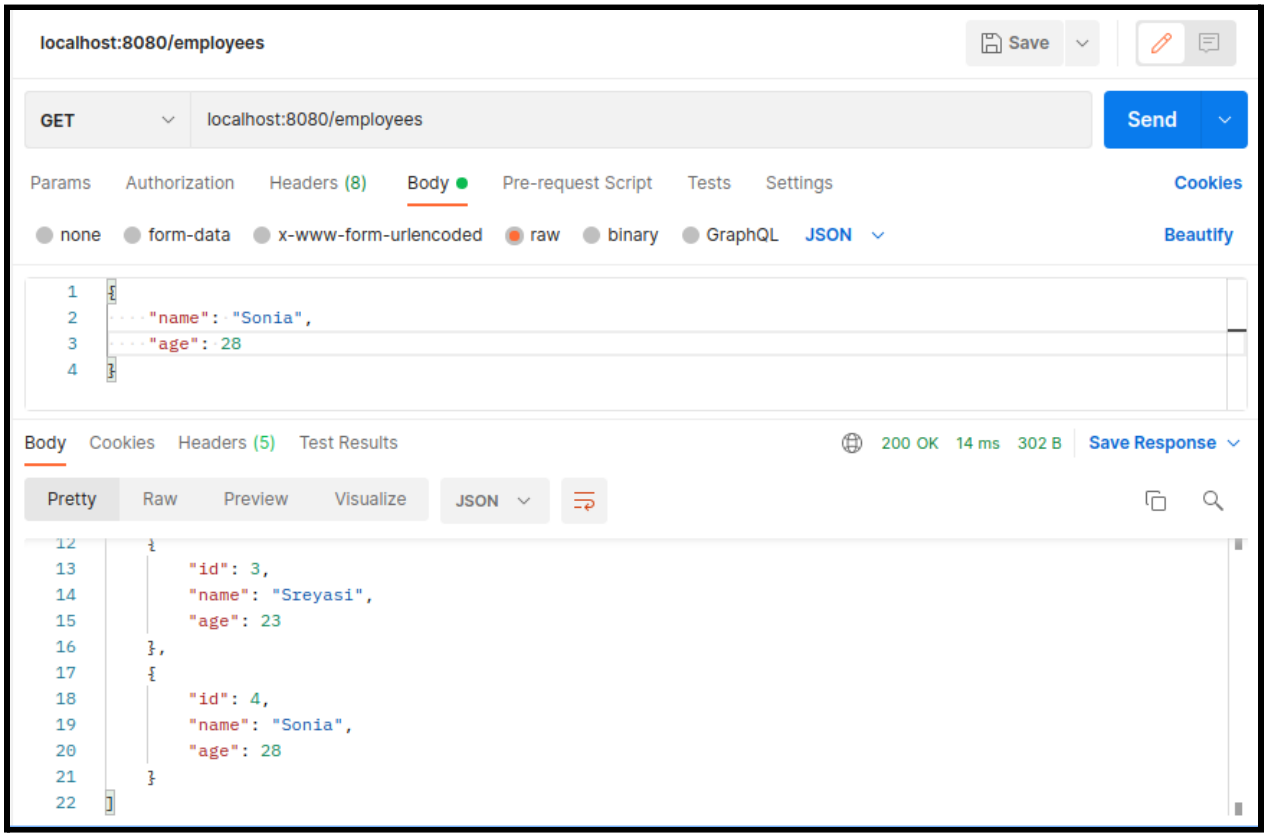
- **Implement POST http request for Employee to create a new employee.**

  <EmployeeController.java>

  @PostMapping("/employees")

  public ResponseEntity<Object> createEmployee(@RequestBody EmployeeBean employee){
  EmployeeBean savedEmployee = service.save(employee);
  URI location = ServletUriComponentsBuilder
  .fromCurrentRequest().path("{id}")
  .buildAndExpand(savedEmployee.getId()).toUri();

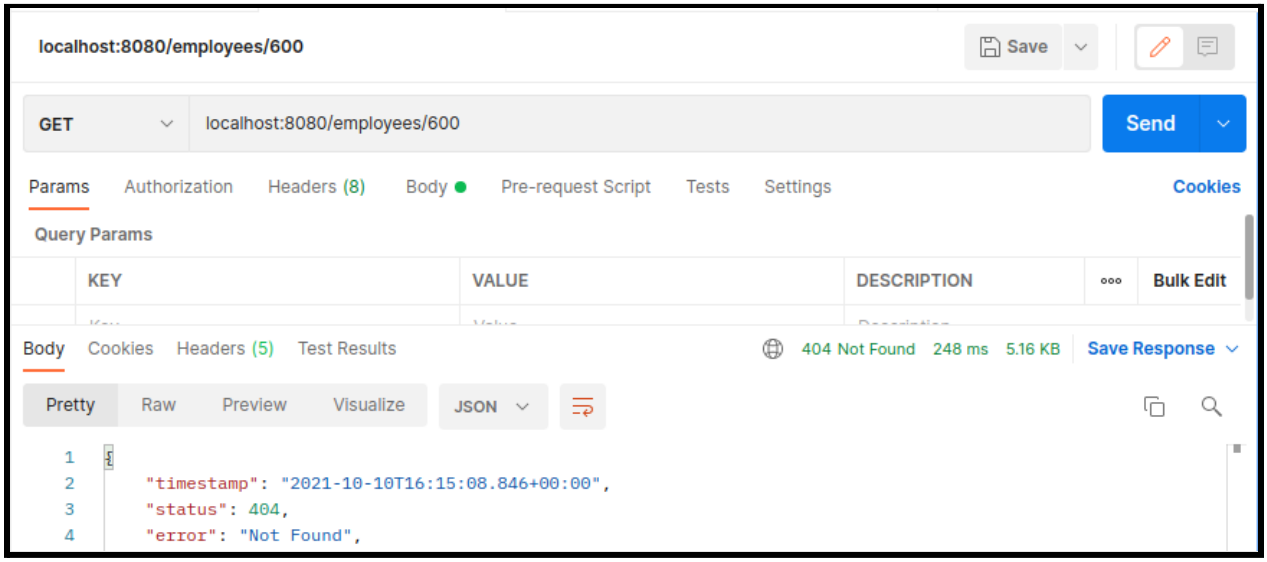
  return ResponseEntity.created(location).build();
  }



- **Implement Exception Handling for resources not found.**

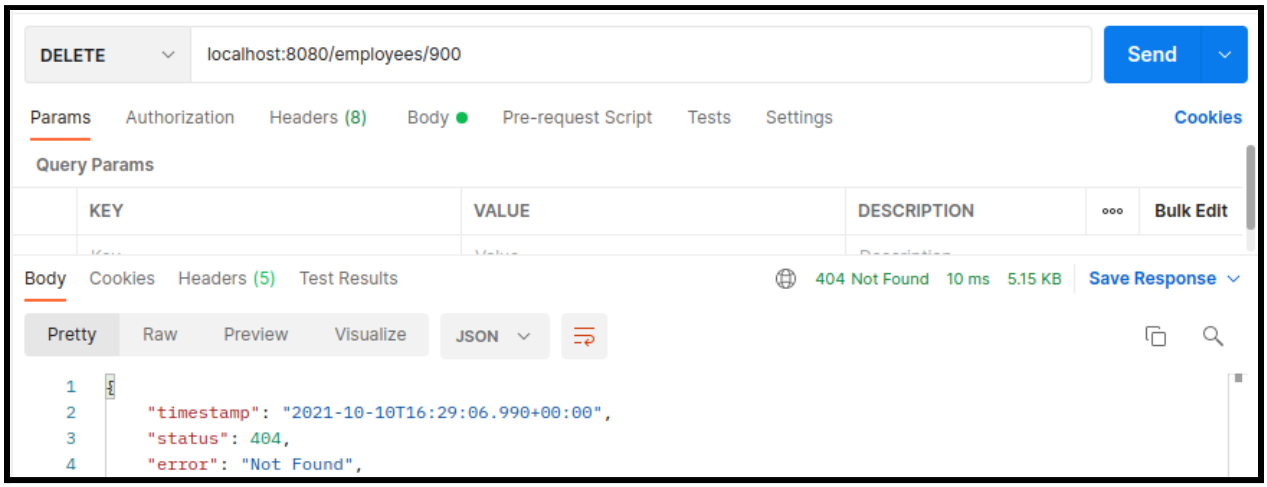
<EmployeeNotFoundException.java>

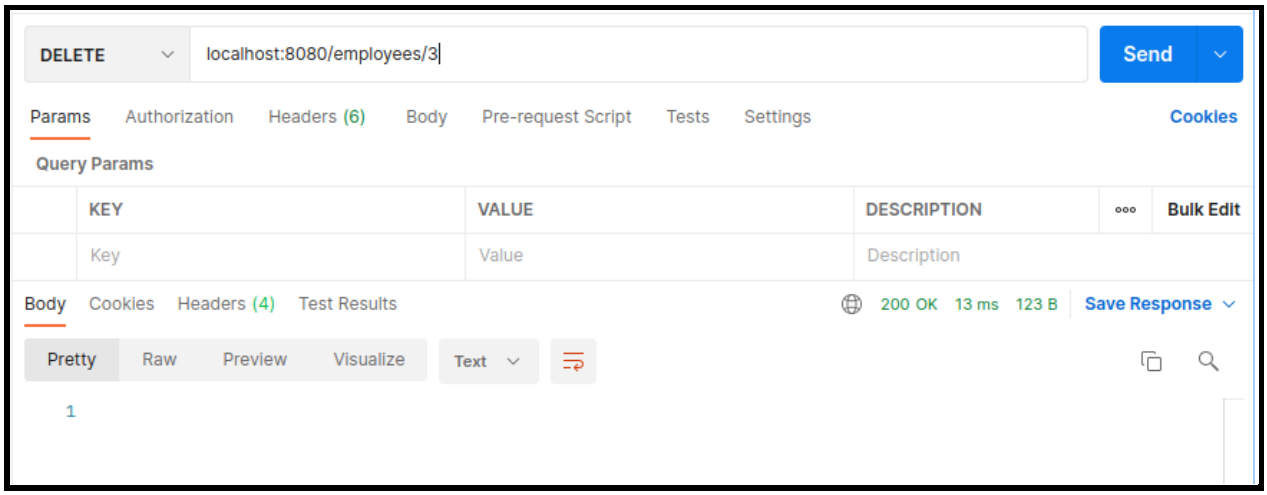
```
package com.ttn.spring.springRest.webServices.RestWebServices1.employee;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(HttpStatus.NOT_FOUND)
public class EmployeeNotFoundException extends RuntimeException{

  public EmployeeNotFoundException(String s){
     super(s);
  }
}
```



- **Implement DELETE http request for Employee to delete employee.**

```
@DeleteMapping("/employees/{id}")
public void DeleteEmployee(@PathVariable int id){
   EmployeeBean employee = service.deleteById(id);
   if(employee == null)
      throw new EmployeeNotFoundException("id-" + id);
}
```





- **Implement PUT http request for Employee to update employee.**

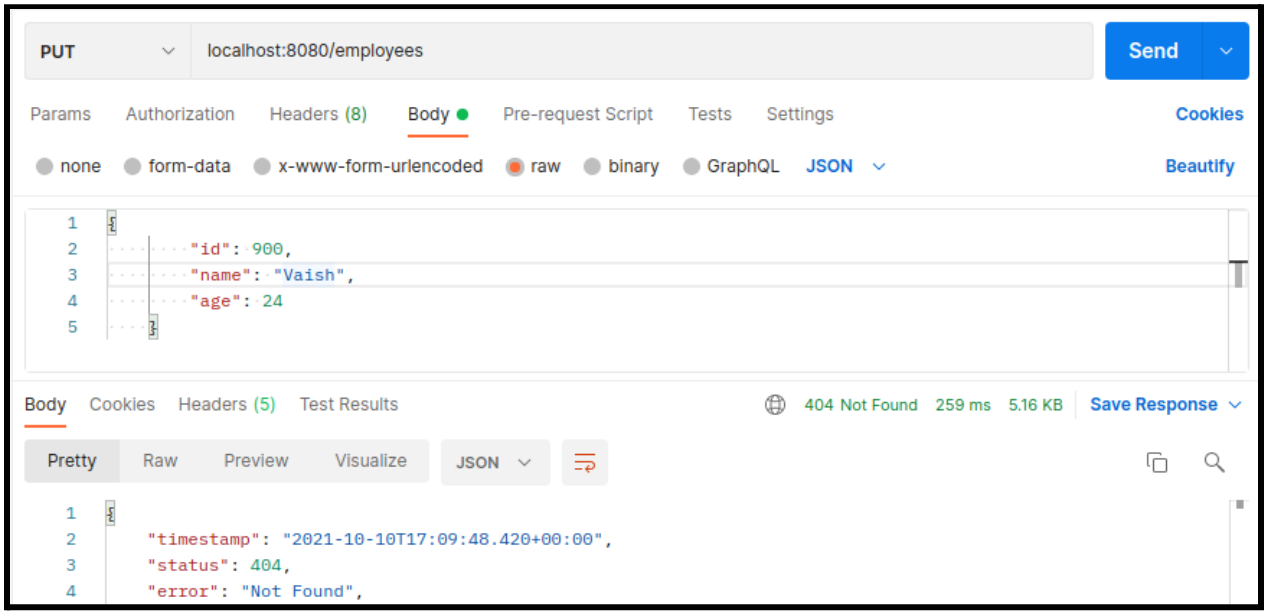
<EmployeeController.java>

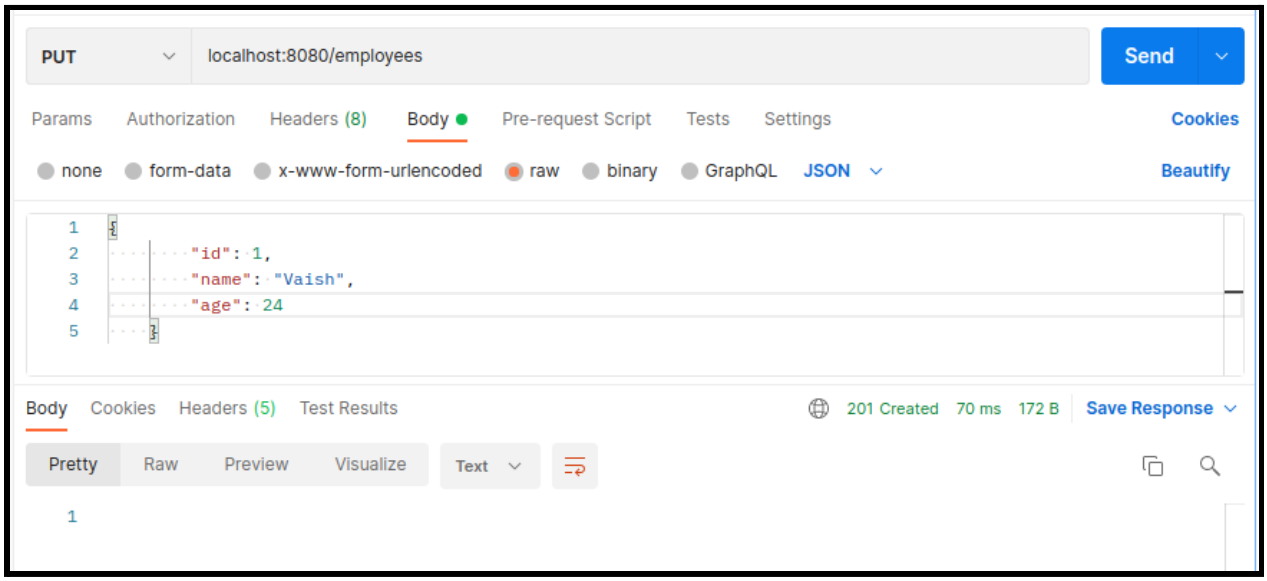
```
@PutMapping("/employees")
public ResponseEntity<Object> updateEmployee(@RequestBody EmployeeBean
employee){
  EmployeeBean updatedEmployee = service.updateEmployeeDetails(employee);
  URI location = ServletUriComponentsBuilder
        .fromCurrentRequest().path("{id}")
        .buildAndExpand(updatedEmployee.getId()).toUri();

  return ResponseEntity.created(location).build();
}
```

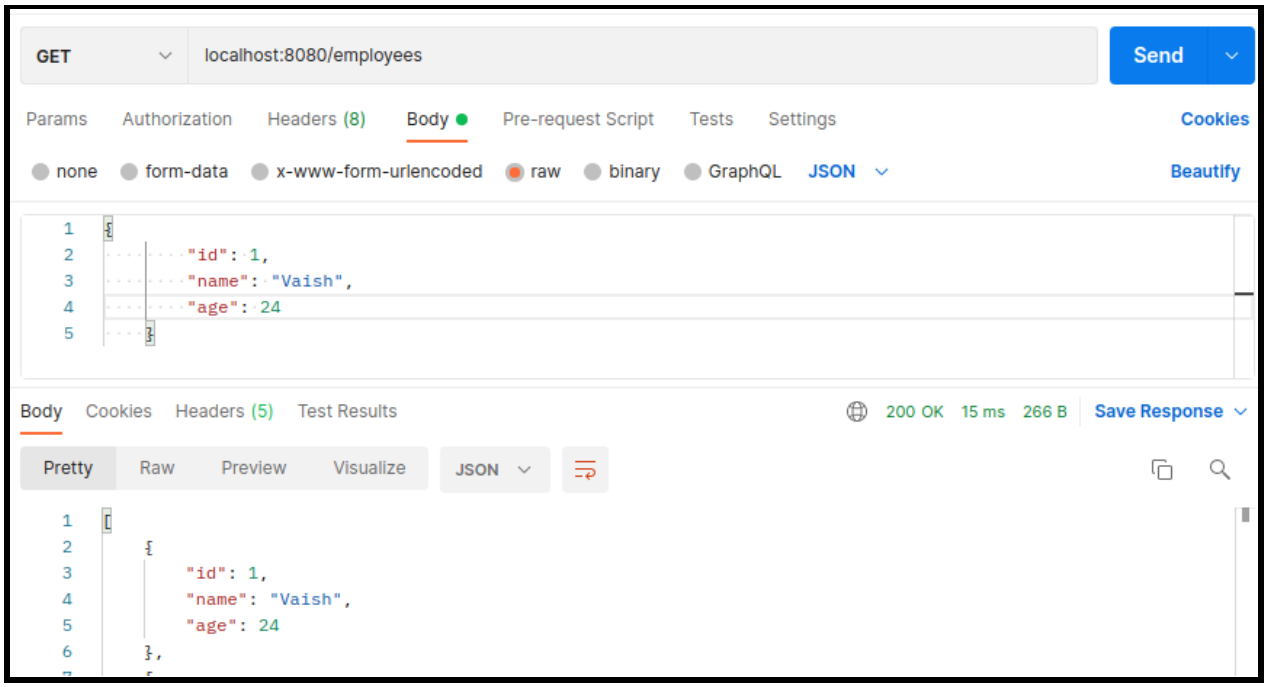
<EmployeeDaoService.java>

```
public EmployeeBean updateEmployeeDetails(EmployeeBean employee){
  for(EmployeeBean emp:employees){
     if(emp.getId() == employee.getId()){
        emp.setName(employee.getName());
        emp.setAge(employee.getAge());
        return emp;
     }
```

```
        }
        return null;
    }
```


PUT
localhost:8080/employees
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"id": 900,
"name": "Vaish",
"age": 24
}
Body
Cookies
Headers (5)
Test Results
404 Not Found
259 ms
5.16 KB
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"timestamp": "2021-10-10T17:09:48.420+00:00",
"status": 404,
"error": "Not Found",


PUT
localhost:8080/employees
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"id": 1,
"name": "Vaish",
"age": 24
}
Body
Cookies
Headers (5)
Test Results
201 Created
70 ms
172 B
Save Response
Pretty
Raw
Preview
Visualize
Text
1



- **Apply validation while create a new employee using POST http Request.**

  <EmployeeBean.java>

```
package com.ttn.spring.springRest.webServices.RestWebServices1.employee;

import javax.validation.constraints.Max;
import javax.validation.constraints.Min;
import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;

public class EmployeeBean {
  private Integer id;

  @Pattern(regexp = "[a-zA-Z]*", message = "Name should contain alphabets only")
  @Size(min = 2, message = "Name length should be greater than 2")
  private String name;

  @Min(value = 18, message = "Minimum age of employee should be 18")
  @Max(value = 60, message = "Maximum age of employee should be 60")
  private Integer age;

  public EmployeeBean(Integer id, String name, Integer age) {
```

```
        this.id = id;
        this.name = name;
        this.age = age;
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "EmployeeBean{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", age=" + age +
                '}';
    }
}
```

<EmployeeController.java>

```
package com.ttn.spring.springRest.webServices.RestWebServices1.employee;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import javax.validation.Valid;
import java.net.URI;
import java.util.List;

@RestController
public class EmployeeController {

  @Autowired
  private EmployeeDaoService service;

  @GetMapping("/employees")
  public List<EmployeeBean> retrieveAllEmployees() {
     return service.findAll();
  }

  @GetMapping("/employees/{id}")
  public EmployeeBean retrieveEmployee(@PathVariable int id){
     EmployeeBean employee = service.findOne(id);
     if(employee == null)
        throw new EmployeeNotFoundException("id- " + id);
     return employee;
  }

  @PostMapping("/employees")
  public ResponseEntity<Object> createEmployee(@Valid @RequestBody
EmployeeBean employee){
     EmployeeBean savedEmployee = service.save(employee);
     URI location = ServletUriComponentsBuilder
           .fromCurrentRequest().path("{id}")
           .buildAndExpand(savedEmployee.getId()).toUri();

     return ResponseEntity.created(location).build();
  }

  @DeleteMapping("/employees/{id}")
  public void DeleteEmployee(@PathVariable int id){
     EmployeeBean employee = service.deleteById(id);
     if(employee == null)
        throw new EmployeeNotFoundException("id-" + id);
```

```
  }

  @PutMapping("/employees")
  public ResponseEntity<Object> updateEmployee(@Valid @RequestBody
EmployeeBean employee){
      EmployeeBean updatedEmployee = service.updateEmployeeDetails(employee);
      if (updatedEmployee == null)
         throw new EmployeeNotFoundException("Incorrect id");
      URI location = ServletUriComponentsBuilder
            .fromCurrentRequest().path("{id}")
            .buildAndExpand(updatedEmployee.getId()).toUri();

      return ResponseEntity.created(location).build();
  }

}
```

<ExceptionResponse.java>

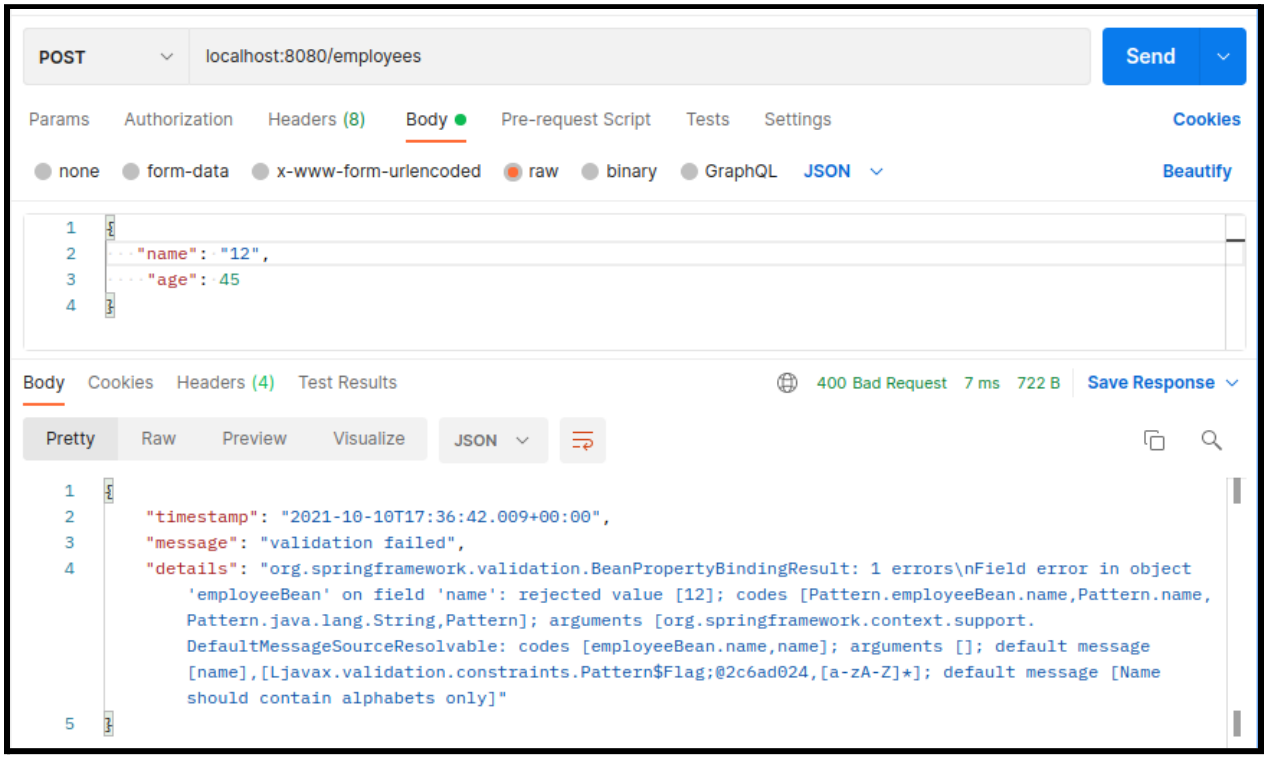
```
package com.ttn.spring.springRest.webServices.RestWebServices1.exception;

import java.util.Date;

public class ExceptionResponse {
  private Date timestamp;
  private String message;
  private String details;

  public ExceptionResponse(Date timestamp, String message, String details) {
     this.timestamp = timestamp;
     this.message = message;
     this.details = details;
  }

  public Date getTimestamp() {
     return timestamp;
  }

  public String getMessage() {
     return message;
  }
```

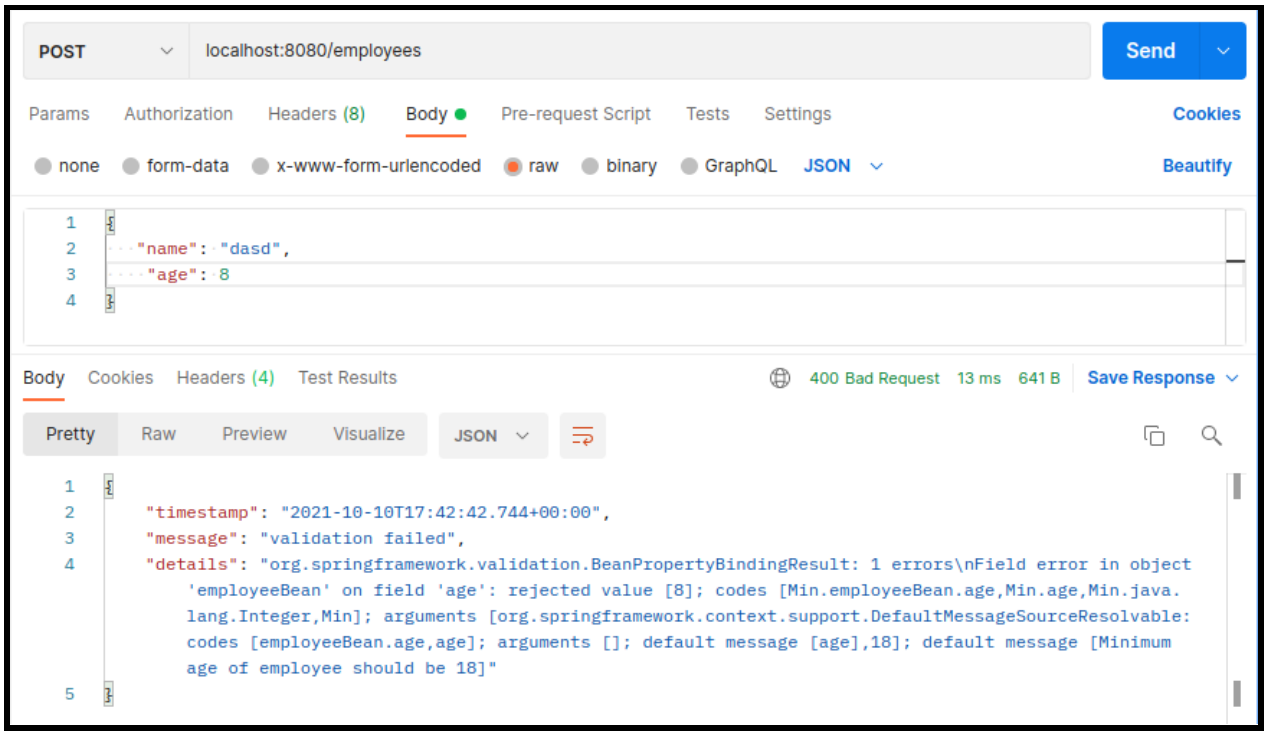
```
  public String getDetails() {
     return details;
  }
}
```

<CustomizedResponseEntityExceptionHandler.java>

```
package com.ttn.spring.springRest.webServices.RestWebServices1.exception;


import
com.ttn.spring.springRest.webServices.RestWebServices1.employee.EmployeeNotFoun
dException;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.context.request.WebRequest;
import
org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHand
ler;

import java.util.Date;

@ControllerAdvice
@RestController
public class CustomizedResponseEntityExceptionHandler extends
ResponseEntityExceptionHandler {

  //For handling all other exceptions
  @ExceptionHandler(Exception.class)
  public final ResponseEntity<Object> handleAllException(Exception ex, WebRequest
request) {
     ExceptionResponse exceptionResponse = new ExceptionResponse(new Date(),
ex.getMessage(), request.getDescription(false));
     return new ResponseEntity(exceptionResponse,
HttpStatus.INTERNAL_SERVER_ERROR);
  }

  //For handling EmployeeNotFoundException
  @ExceptionHandler(EmployeeNotFoundException.class)
```

```
    public final ResponseEntity<Object> handleUserNotFoundException(
            EmployeeNotFoundException ex, WebRequest request) {

        ExceptionResponse exceptionResponse = new ExceptionResponse(
                new Date(), ex.getMessage(), request.getDescription(false));

        return new ResponseEntity(exceptionResponse, HttpStatus.NOT_FOUND);
    }

    @Override
    protected ResponseEntity<Object> handleMethodArgumentNotValid(
            MethodArgumentNotValidException ex, HttpHeaders headers, HttpStatus status,
            WebRequest request) {

        ExceptionResponse exceptionResponse = new ExceptionResponse(
                new Date(),"validation failed", ex.getBindingResult().toString());

        return new ResponseEntity(exceptionResponse, HttpStatus.BAD_REQUEST);

    }
}
```





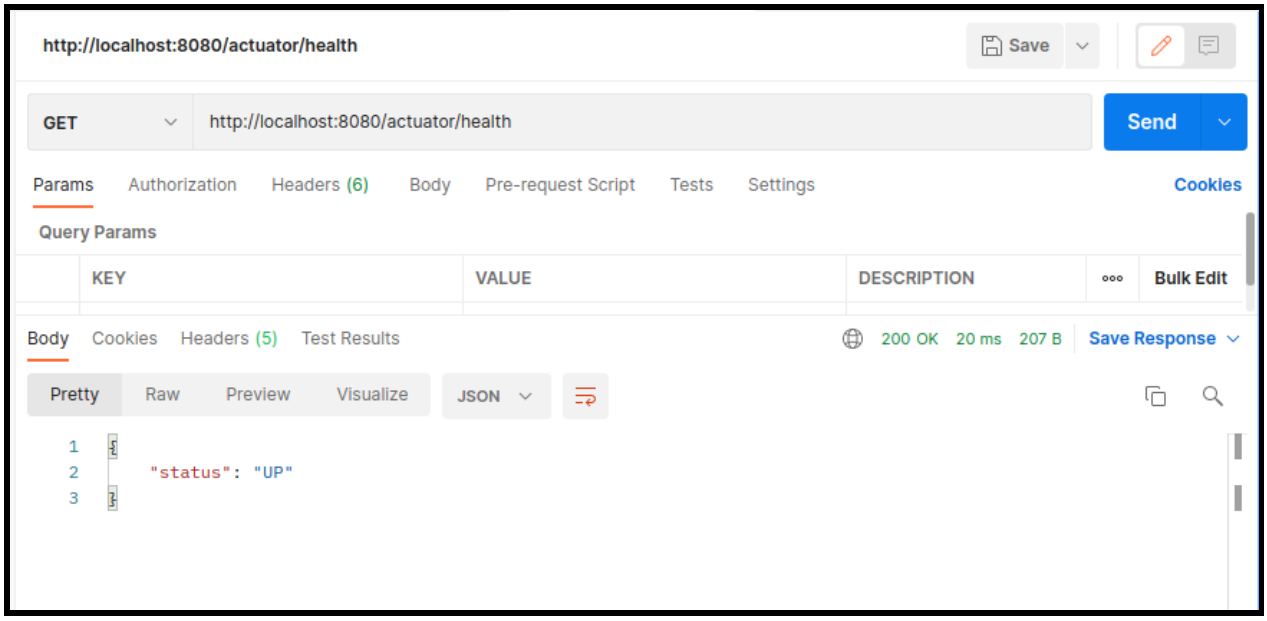
- **Configure actuator in your project to check the health of application and get the information about various beans configured in your application.**

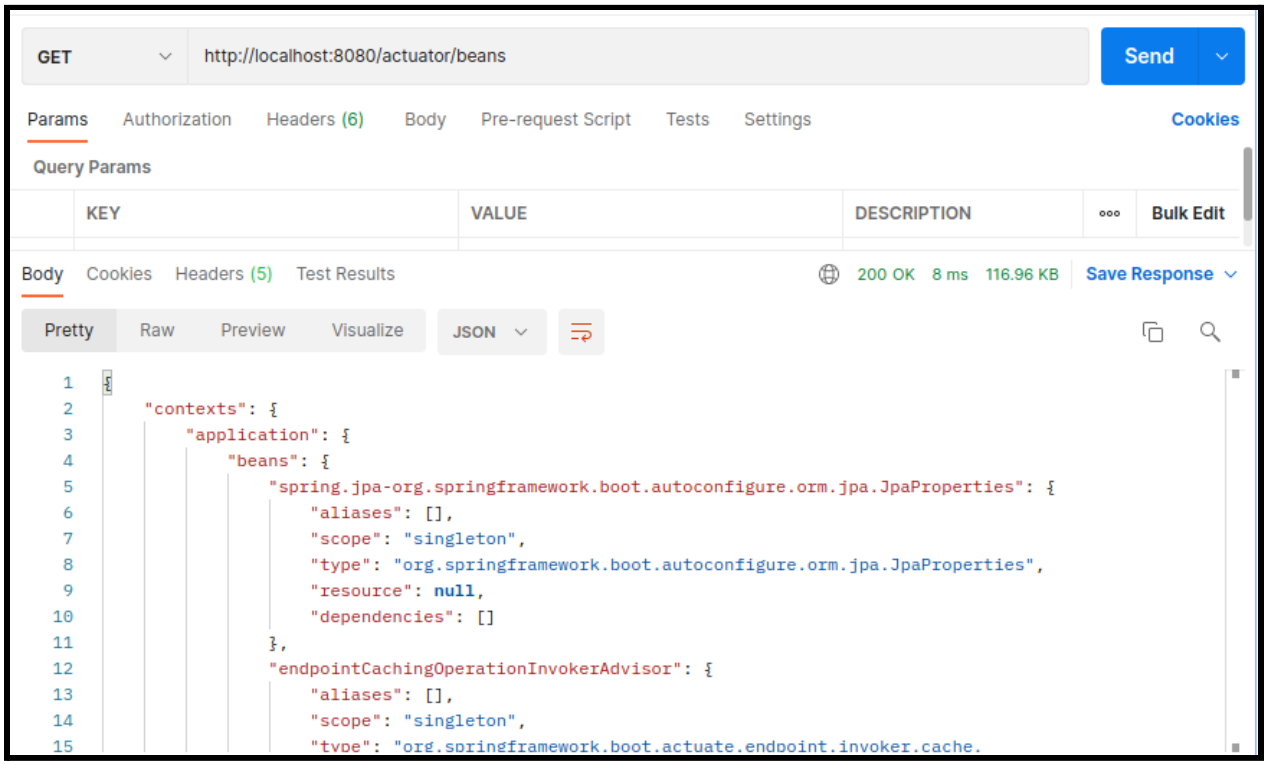
  <application.properties>

  logging.level.org.springframework = debug
  management.endpoints.web.exposure.include=*

  <Pom.xml>

  <dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
  </dependency>


http://localhost:8080/actuator/health
Save
GET
http://localhost:8080/actuator/health
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Body
Cookies
Headers (5)
Test Results
200 OK
20 ms
207 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"status": "UP"
}


GET
http://localhost:8080/actuator/beans
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Body
Cookies
Headers (5)
Test Results
200 OK
8 ms
116.96 KB
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"contexts": {
"application": {
"beans": {
"spring.jpa-org.springframework.boot.autoconfigure.orm.jpa.JpaProperties": {
"aliases": [],
"scope": "singleton",
"type": "org.springframework.boot.autoconfigure.orm.jpa.JpaProperties",
"resource": null,
"dependencies": []
},
"endpointCachingOperationInvokerAdvisor": {
"aliases": [],
"scope": "singleton",